FINAL
00-2150 12 pp

Deep Space Network Turbo Decoder Implementation[1]

Jeff B. Berner
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-3934
jeff.b.berner@jpl.nasa.gov

Kenneth S. Andrews
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-5865
kenneth.s.andrews@jpl.nasa.gov

Abstract— A new decoder is being developed by the Jet Propulsion Laboratory for NASA's Deep Space Network. This unit will decode the new turbo codes, which have recently been approved by the Consultative Committee for Space Data Systems (CCSDS). Turbo codes provide up to 0.8 dB improvement in E_b/N_0 over the current best codes used by deep space missions.

The new decoder is being implemented in software running on commercial Digital Signal Processor (DSP) chips, removing the need to design complicated and expensive hardware as was the case with the previous generation of codes. The decoder will time-tag the data frames, perform frame synchronization in the symbol domain (as opposed to the current bit domain synchronization), decode the turbo coded frames, and output the decoded bits in the CCSDS Standard Formatted Data Units format. The decoder is initially designed to operate up to 365 kbps, but will increase in rate as DSP clock rates increase. The implementation will go operational in October, 2003.

TABLE OF CONTENTS

1. INTRODUCTION
2. DESCRIPTION OF TURBO CODES
3. ADVANTAGES OF TURBO CODES
4. TURBO ENCODING
5. TURBO DECODING
6. USAGE ISSUES
7. IMPLEMENTATION STATUS
8. CONCLUSION
9. REFERENCES
10. ACKNOWLEDGEMENTS

1. INTRODUCTION

In 1994, a new class of error correcting codes was developed [1]. These codes provide up to an 0.8 dB improvement over the current best codes used by deep space missions. The Consultative Committee for Space Data Systems (CCSDS) has recommended a set of these codes for use in future missions.

The Jet Propulsion Laboratory (JPL) is developing a turbo decoder for use in the Deep Space Network (DSN) that JPL manages for NASA. This decoder not only decodes the turbo encoded data, but also performs time-tagging and frame synchronization as part of the decoding process. As opposed to previous decoder implementations, this decoder is implemented in software that runs on commercial Digital Signal Processor (DSP) chips. This provides both a quicker development time and a cheaper production implementation.

This paper is divided into four parts. First, we describe turbo codes and how they are used. Next, we show the advantages of turbo codes versus the currently used deep space error correcting codes. We then briefly discuss the options for implementing the encoding on the spacecraft. Finally, we describe the design and implementation of the turbo decoder that is being developed by JPL.

2. DESCRIPTION OF TURBO CODES

Turbo codes are block codes. That is, the encoding is done on one block of data at a time. A transfer frame (as defined by [2] and shown in Figure 1) is the basic block. As part of the transfer frame, the 16-bit Frame Error Control Field (FECF) at the end of the frame is required; the FECF is a Cyclic Redundancy Code (CRC). After the encoding is done, the frame synchronization marker is attached to the

[1] 0-7803-6599-2/01/$10.00 © 2001 IEEE

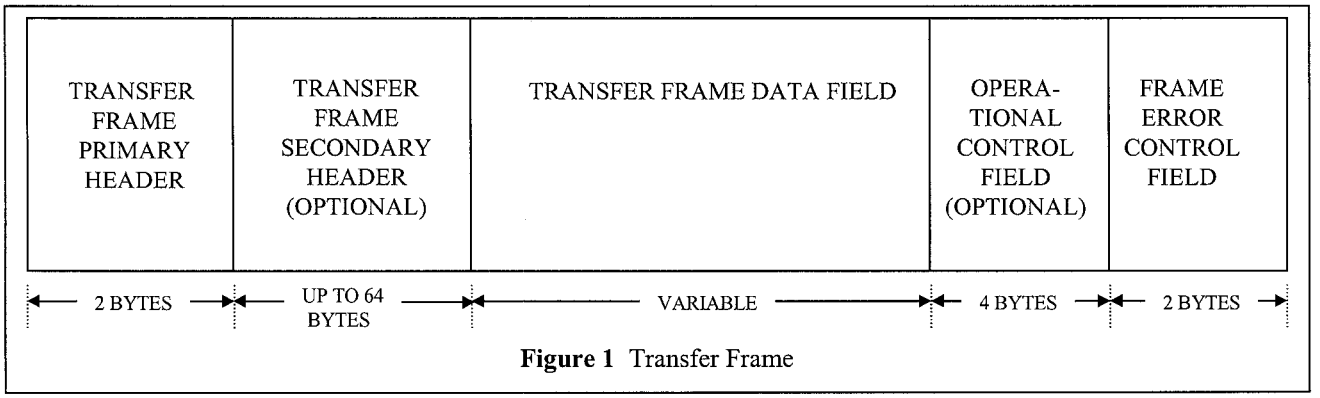


Figure 1 Transfer Frame

beginning of the block. This is different than the sequence for the concatenated convolutional / Reed-Solomon encoding that is currently done for deep space systems. In that case, the transfer frame is Reed-Solomon encoded (which is a block code), has the frame synchronization marker appended, and then is convolutionally encoded. Figure 2 compares the two sequences.

16-bit CRC that is used after the turbo decoding to detect remaining bit errors. When used as a frame "goodness" indicator, the probability of an erroneous frame being accepted as a good frame is very small (approximately the BER divided by 2^{16}, or 1.5×10^{-5} BER). The next revision of the turbo encoding recommendation [3] will recommend that the FECF always be used with turbo encoding.

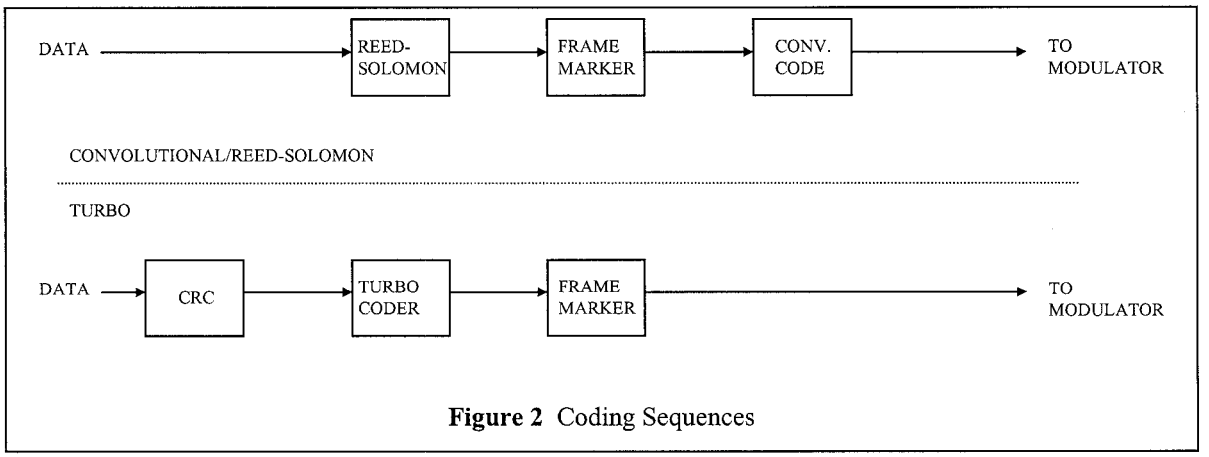


Figure 2 Coding Sequences

The stages in generating the output data stream are described below.

2.1 Frame Error Control Field

One feature of the Reed-Solomon coding is its use as a final "good frame/bad frame" indicator. The decoder either corrects the frame or indicates that it cannot decode the frame. The probability of its making an error in that process is very small (approximately the Bit Error Rate (BER) of the code divided by 16!, or 4.8×10^{-14} BER), so it can be used as a reliable frame "goodness" indicator.

Unfortunately, by itself, a turbo code does not have that property. Its error floor on the decoded frame is higher than what the Reed-Solomon code provides. However, when using the optional FECF, as defined in the Transfer Frame specification [2], this problem is alleviated. The FECF is a

2.2 Encoding Algorithm

Turbo encoding is very straightforward. It uses two constraint length 4 convolutional codes to generate the encoded symbols. The process is as follows:

The transfer frame is input to the encoder. There are four frame sizes defined: 1784, 3568, 7136, and 8920 bits (a fifth frame size, 16384 bits, is defined, but the encoding parameters have yet to be specified). Note that the frame sizes correspond to the Reed-Solomon frame's data allocation (the same amount of data is sent per frame). The frame of data (of length k bits) is input to one of the convolutional encoders. An interleaved (permuted) version of the data frame is input to the second encoder. For a code rate of $1/n$, the original bit and n-1 coder tap outputs are output as the encoded symbols (this means that the unencoded bits are available, if necessary). The CCSDS

recommendation defines turbo codes for n equal to 2, 3, 4, and 6. After the k bits are clocked into the encoders, four flush bits are input; these flush bits clear out the coder memory (also known as trellis termination). Thus, for a k bit frame, we get $n(k+4)$ symbols output. Figure 3 shows the turbo encoder structure and Figure 4 shows the interleaving concept.

The interleaving is a fixed sequence, which is on a bit-by-bit level (as opposed to the Reed-Solomon interleaving on a byte-by-byte level). The interleaving is described algorithmically below (as excerpted from [3]):

> First express k as $k=k_1k_2$, where k_1 equals 8.
>
> Next do the following operations for $s=1$ to $s=k$ to obtain permutation numbers $\pi(s)$, where s indicates the sth bit input to the second (interleaved) encoder and $\pi(s)$ is the bit number in the original frame. In the equation below, $\lfloor x \rfloor$ denotes the largest integer less than or equal to x, and p_q denotes one of the following eight prime integers:
>
> $$p_1 = 31$$
> $$p_2 = 37$$
> $$p_3 = 43$$
> $$p_4 = 47$$
> $$p_5 = 53$$
> $$p_6 = 59$$
> $$p_7 = 61$$
> $$p_8 = 67$$
>
> $$\begin{aligned} m &= (s-1) \bmod 2 \\ i &= \left\lfloor \frac{s-1}{2k_2} \right\rfloor \\ j &= \left\lfloor \frac{s-1}{2} \right\rfloor - i k_2 \\ t &= (19i+1) \bmod \frac{k_1}{2} \\ q &= t \bmod 8 + 1 \\ c &= (p_q j + 21m) \bmod k_2 \\ \pi(s) &= 2\left(t + c\frac{k_1}{2} + 1\right) - m \end{aligned}$$

The symbol outputs are shown in Figure 3. For every input (either an input bit or a flush bit) to the shift registers, n symbols are output. The output sequence is from top to bottom in the figure (e.g., for rate 1/6, the output sequence is 0a, 1a, 2a, 3a, 1b, 3b).

2.3 Pseudo Randomization

If sufficient data transitions (which are required to lock up the receiving system) are not guaranteed, either by the modulation scheme or the data stream, then the CCSDS recommends that a Pseudo Randomizer be used on the encoded data. This means that a pseudo noise (PN) sequence is exclusive-ORed bit by bit with the encoded data. The PN sequence that is used is a 255-bit length sequence, which repeats as it is cycled through the encoded block. The sequence is defined by the following generator polynomial:

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1$$

The sequence generator is always initialized to the all-ones state for the beginning of each encoded block.

Although it is currently an option, it is expected that, in the next revision of the standard, pseudo randomization will be required.

2.3 Frame Synchronization Marker

Once the frame is encoded, a frame synchronization marker must be applied. This is similar in concept to what is currently done with the Reed-Solomon encoding. The only difference here is that the encoded block is already in the symbol domain, so the frame marker must be applied in the symbol domain. Thus, the 32-bit marker is appended as a 32n-symbol marker (e.g., 96 symbols for rate 1/3, 192 symbols for rate 1/6). The markers for the different code rates are defined in [3]. If pseudo randomization is used, the synchronization marker is not exclusive-ORed with the PN sequence.

3. ADVANTAGES OF TURBO CODES

There are three main advantages to using turbo codes: a performance gain when compared to the other codes, compliance with the CCSDS, and a reduction in the decoder complexity. Each is discussed below.

3.1 Performance Gain

The performance of a code can be judged by the ratio of the energy-per-bit to the noise spectral density (E_b/N_0) needed to achieve a desired probability of error (P_e); P_e is also known as the Bit Error Rate (BER). The lower the E_b/N_0 required for a given P_e, the better the code's performance.

Figures 5 and 6 provide a comparison of the two standard codes currently used for deep space and two of the new turbo codes. The current codes are the constraint length 7, rate 1/2 convolutional code (denoted as the (7, 1/2) code), concatenated with the (255, 223) Reed-Solomon code, and the constraint length 15, rate 1/6 (15, 1/6) convolutional code, also concatenated with the (255, 223) Reed-Solomon code. The turbo codes plotted are the rate 1/3 and rate 1/6 codes. Figure 5 compares the performance for frame sizes of 1784 bits, which corresponds to a Reed-Solomon code interleave factor of 1. Figure 6 provides the comparison for a frame size of 8920 bits, which corresponds to a Reed-Solomon interleave factor of 5. As can be seen, for the long frame at a BER of 10^{-6}, the rate 1/6 turbo code provides approximately 0.8 dB improvement over the (15, 1/6) concatenated code and the rate 1/3 code provides a 0.4 dB improvement. Comparing the (7, 1/2) code with the two

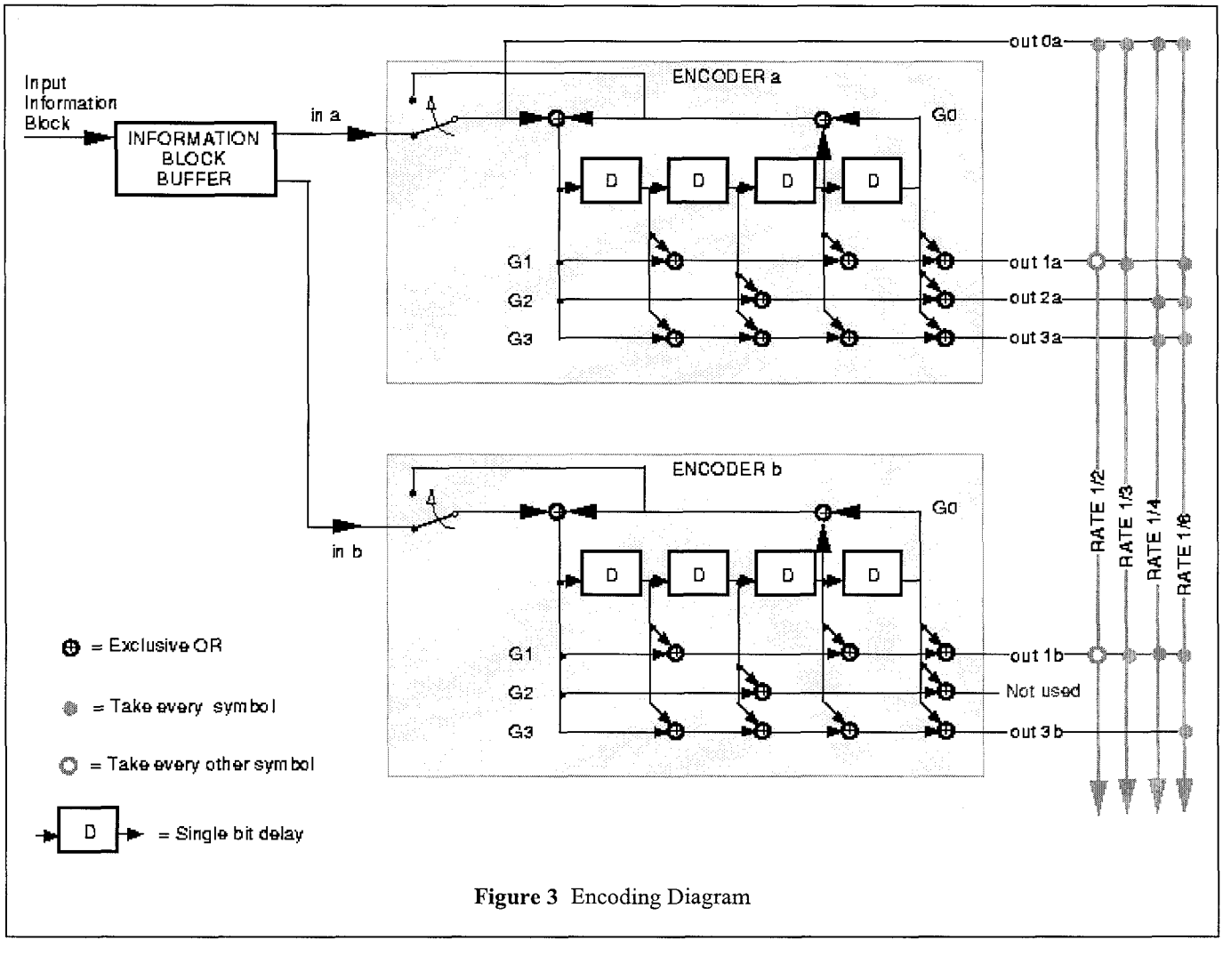


Figure 3 Encoding Diagram

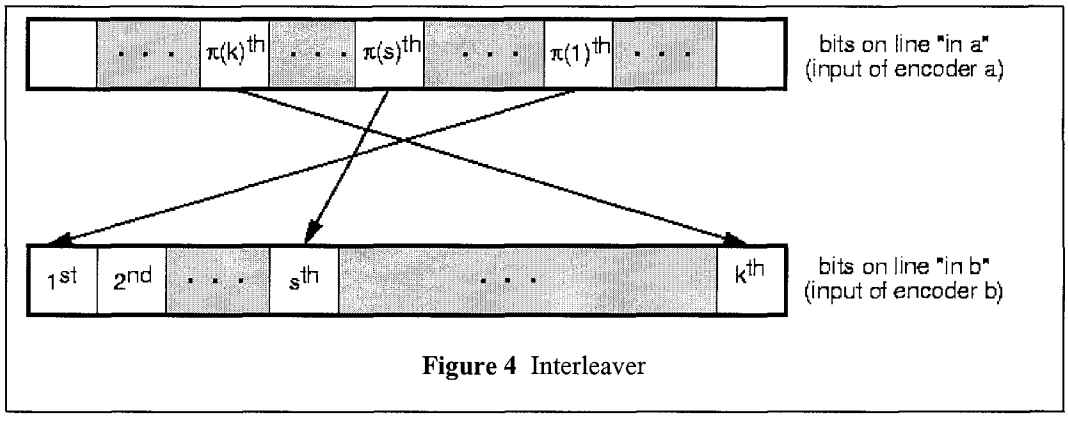


Figure 4 Interleaver

turbo codes, we see a 2.7 dB improvement for the rate 1/6 turbo code and a 2.2 dB improvement for the rate 1/3.

3.2 CCSDS Compliance

The CCSDS provides recommendations for telemetry processing. By using a code that is approved by the CCSDS, the possibility of inter-Agency cross support is enhanced. For example, the European Space Agency (ESA) is building a turbo decoder to support the Rosetta mission. Any ESA antenna that is used to support Rosetta could be used to support NASA missions that use turbo codes. On the other hand, the (15, 1/6) codes were never included in the standard by the CCSDS and there are no antennas outside of the DSN antennas that support them.

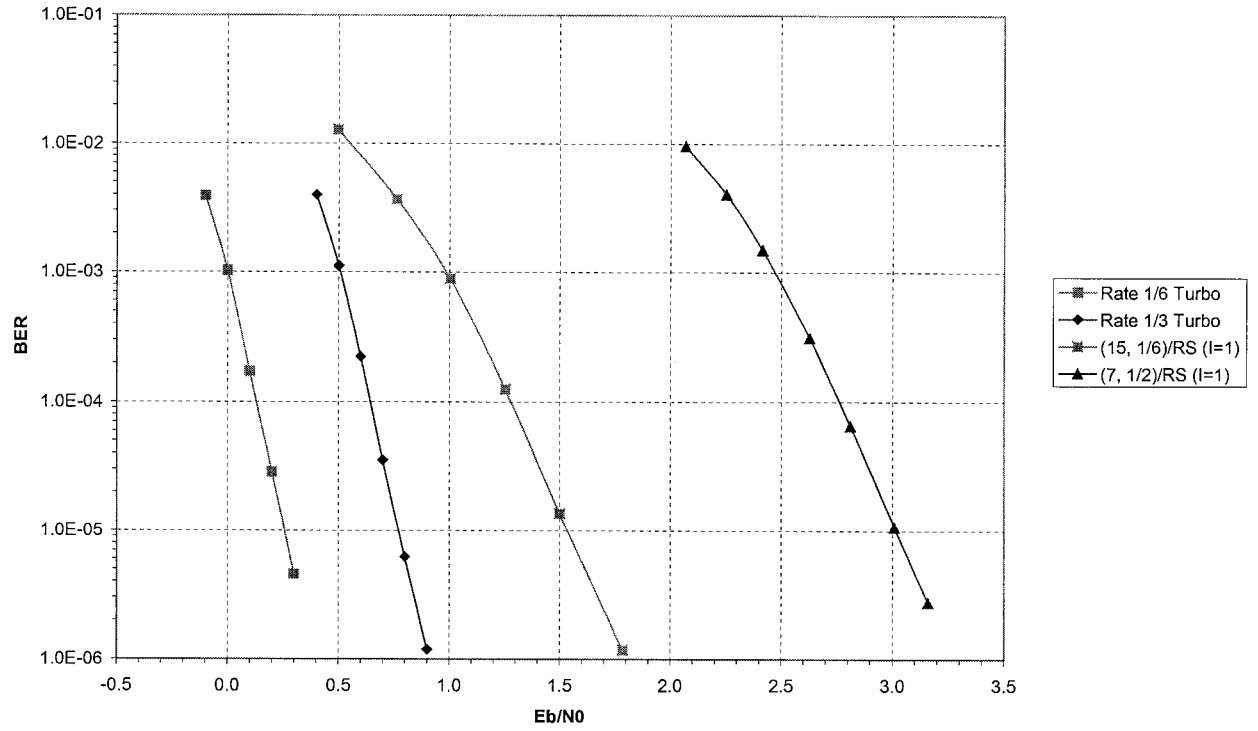


Figure 5 Code Performance (1784 Bit Frames)

3.3 Decoder Complexity

The complexity of the decoder for convolutional and turbo codes can be estimated by the number of trellis edges which must be evaluated per output bit. The (15, 1/6) convolutional code has 2^{14} states and 2^{15} edges per trellis section. When decoded with the Viterbi algorithm by the DSN's MCD III, each edge is involved once in the Add-Compare-Select operations, so we assign this system a complexity of $2^{15} = 32768$.

The CCSDS turbo codes are constructed from a pair of convolutional codes, each with 2^5 edges per trellis section. Turbo decoding uses the more complex Bahl-Cocke-Jelinek-Raviv, or "forward-backward" algorithm [4], which involves each trellis edge three times. This gives a complexity of 96 per half-iteration, so a turbo decoder which performs ten iterations on every block has a complexity of 1920.

By this measure, decoding a CCSDS turbo code is about 17 times less complex than decoding the (15, 1/6) convolutional code. A more accurate complexity comparison can be made when the decoder technology is specified. When using Digital Signal Processors, the "forward-backward" computations are nearly twice as complex as the Viterbi computations, so the true complexity reduction is about a factor of 10.

What this means is it is simpler to build a turbo decoder than a decoder for the (15, 1/6) code. This is what allows the turbo decoder to be implemented as software on commercial DSP boards, instead of the costly custom Application Specific Integrated Circuit (ASIC) implementation that was required for the MCD III.

4. TURBO ENCODING

Obviously, to take advantage of the turbo codes, the encoding function must be implemented on the spacecraft. Below we talk about the several options that a spacecraft implementation can chose from.

4.1 Spacecraft Transponding Modem

The Spacecraft Transponding Modem (STM) is a new transponder that has been developed at JPL for use by missions after 2003 [5]. Among its features is a frame level telemetry interface. The STM performs all of the encoding, including turbo encoding. Two of the rates are supported, 1/3 and 1/6, and two of the frame sizes are supported, 1784 and 8920. Any mission that uses the STM automatically gets turbo encoding capability, along with the other codes (convolutional and Reed-Solomon).

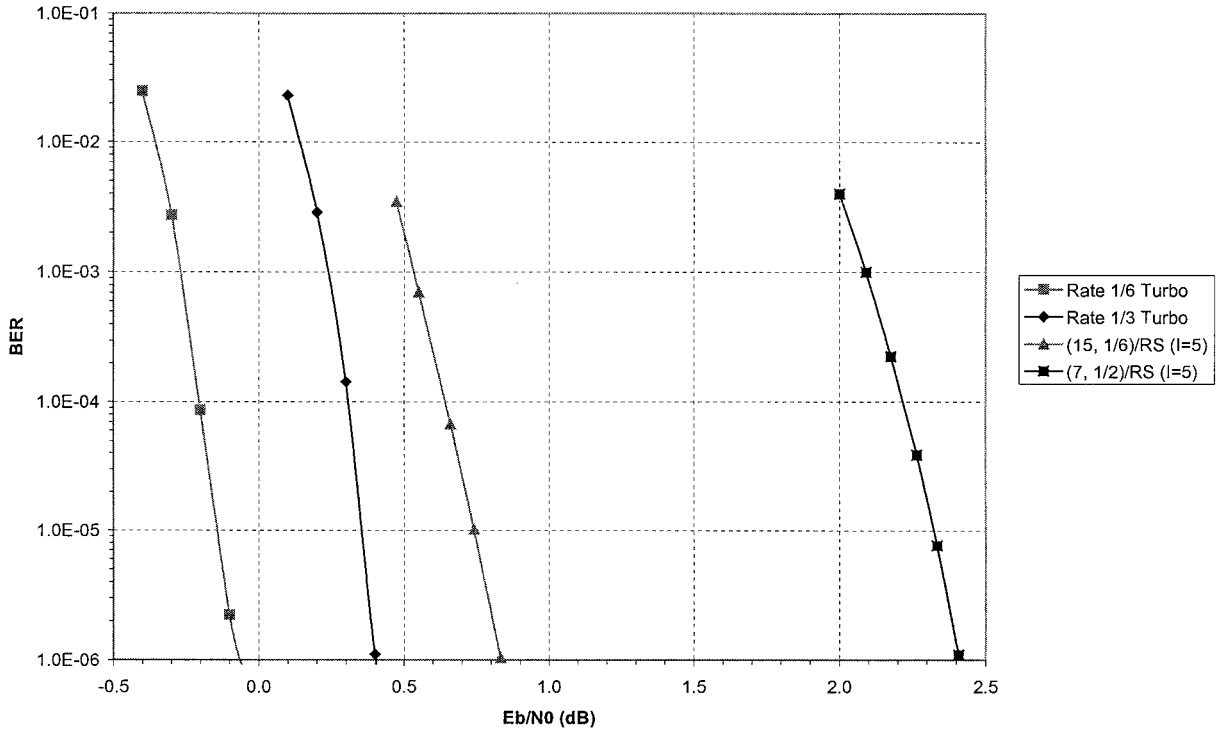


Figure 6 Code Performance (8920 Bit Frames)

One caveat must be mentioned about the current version of the STM's turbo implementation. When the STM digital ASIC was designed, the CCSDS had not yet reached a consensus on the code implementation, so the developers used what was the current proposal. Afterwards, the code definition changed slightly, making the STM non-compliant with the CCSDS (though this is not a problem for the turbo decoder that will be installed in the DSN). It is expected that when the ASIC is implemented in a radiation hardened process, this problem will be corrected.

4.2 Software

The turbo encoding algorithm is relatively straightforward to implement in software on a spacecraft. All there is to it are the algorithms for the interleaving and the shift registers. C-language software has been developed and is available for use. It is hoped that this software, or a version of it, will be made available in the future by the CCSDS, as a method for validating an encoder implementation.

4.3 Hardware

The turbo encoding function can also be implemented in hardware, either in a Field Programmable Gate Array (FPGA) or in an ASIC. The function can be implemented as a stand-alone add-on to a system, or as a part of an integrated telemetry solution.

For example, the X2000 project at JPL is implementing an interface assembly, called the System Interface Assembly (SIA), for interfacing from the PCI bus to either the STM or the Motorola Small Deep Space Transponder (SDST). Since the SDST does not have turbo encoding capability, the SIA will have the encoding function included (also included will be the Reed-Solomon encoding - the SDST provides the convolutional encoding); in other words, using the SIA, the SDST string will have a frame level telemetry interface like the STM interface. The SIA implements the turbo encoding function as part of a general purpose telemetry processing ASIC.

5. TURBO DECODING

On the receiving side, the turbo decoder takes 8-bit quantized symbols from the receiver and produces the decoded bits, along with time-tag information. The process is described below.

5.1 Description

The turbo decoder actually encompasses several functions: time tagging, frame synchronization, pseudo derandomization, turbo decoding, and CRC checking. More detail is available in [6].

Time tagging is accomplished by counting the cycles of the 10 MHz reference signal, using a 1 pulse per second (pps) timing reference to zero out the count (the 1 pps occurs on the second boundary). Each symbol clock, which clocks the symbol into the decoder, latches this count. This process gives a 24-bit count of the 0.1μsec of the current second. The I/O processing of the decoder adds the current second to the count. Thus, each symbol has a time tag associated with it.

Next, frame synchronization is performed. The frame synchronizer searches for the frame synchronization marker that was appended to the coded block. The frame synchronizer checks for both normal and inverted polarity in the marker; if it detects the inverted polarity, the encoded block is marked for inversion before being sent to the decoding task. The synchronizer can buffer a minimum of four frames. This allows the system to acquire synchronization and then apply it backwards to the previous frames, reducing loss of data during the lock up period. Also, due to the fact that the synchronization search is done in the symbol domain (as opposed to the bit domain as has always been done for convolutional/Reed-Solomon coding), the SNR that the synchronizer operates at is lower; this requires that multiple frames be summed to achieve high enough SNR for determining synchronization.

The synchronized block is passed to the decoder element, along with a flag indicating whether or not the block is inverted. Unlike the convolutional/Reed-Solomon codes, turbo codes are not transparent; the inversion of a codeword is not a codeword. So, if indicated, the decoder must invert the block prior to decoding. Also, if pseudo randomization was applied to the codeblock, it must be removed.

The turbo decoder itself is an iterative decoder. Each of the two codes (non-interleaved and interleaved) are alternately decoded, with the results being passed back and forth between the two. Eventually, the decoder produces an output. The decision on when to stop iterating can be achieved either by performing a fixed number of iterations or by using a metric to determine that the decoder has converged to a result. These methods are described in the next section.

Since the decoder operates on blocks (frames), higher speed can be achieved by having multiple decoder elements. The total speed of the decoder is the product of the number of decoder elements and the speed of an individual element. The current requirement for the first turbo decoder implementation is a 365 kbps rate.

The turbo decoder is implemented on commercial Digital Signal Processor (DSP) boards. There are two boards, each with four Texas Instruments (TI) TMS320C6000 family DSPs, giving a total of eight DSPs for the task. The design provides a resource allocation of one DSP to handle the high rate symbol input and time tagging, one to handle the frame synchronization, one to handle the control of passing the blocks to and from the decoder elements and to the system output, and five DSPs for the actual decoder elements. However, the prototype development indicates that the symbol input, frame synchronization, and control processing only require a total of two DSPs, allowing for a sixth DSP to be allocated for decoding.

5.2 Decoder Speed

Decoder speed has several potential bottlenecks: the symbol input, the frame synchronizer, and the decoder elements. All of the processing are functions of the DSP clock speed. Current development is on 200 MHz boards; all of the numbers quoted in this section are for these boards.

Input rates of 16 MHz have been successfully demonstrated. This would give a symbol input rate of 16 Msps, which translates to 2.67 Mbps for a rate 1/6 code. It is expected that higher speed processors will be able to handle higher rates, up to a maximum input rate of 26.4 Msps (the maximum output rate of the receiver). However, even at the current speed, the symbol input will not be a bottleneck.

The frame synchronizer must be run in serial with the data stream (as opposed to the parallelism that can be achieved with the decoder elements). Current projections from the prototype indicate that it runs at a rate of 4.5 Msps for the worst case (192 symbol frame marker and an 8920 bit frame). This corresponds to a bit rate of 750 kbps for a rate 1/6 code. This will increase with faster speed processors. In addition, if a higher throughput is needed, there are some additional tricks that can be done for higher symbol rates.

The decoder elements are the main bottlenecks in the speed equation. There are three ways to increase the speed of the decoding. First, the processor speed can be increased; the decoder speed is basically linear with the processor speed (doubling the speed doubles the decoding rate). This is definitely a viable option; as mentioned earlier, while the current development has been done on 200 MHz DSPs, 300 MHz DSPs are available, and TI has promised a 1 GHz DSP in the future. Secondly, more DSPs can be added to the system.

The final way to increase the decoding rate is to use stopping rules in the decoding process. As described earlier, the decoder is an iterative process that can run for a fixed number of iterations, or can be stopped, when convergence is detected. The method for determining the convergence is called the stopping rule. Stopping rules are a method of determining, after every iteration, whether or not the decoder has converged to a solution. Once the decoder converges, the block is output and a new block is accepted for processing. On the average, this is expected to increase the overall average speed of the decoder by about a factor of two, versus using a fixed number of iterations (currently, 10 iterations are done). These stopping rules are currently being investigated [7].

6. USAGE ISSUES

In addition to the implementation questions, there are other issues that affect how a mission would use turbo codes. They are discussed below.

6.1 Loss Models

Before the symbols reach the turbo decoder, they are demodulated from a carrier and subcarrier. These processes have losses that degrade the signal. Models for these losses are being developed. Preliminary results have been published [8] and been incorporated in [9]. These models allow a mission to design a link using turbo codes. The work is continuing.

6.2 Data Output

The data blocks are output to the project as Standard Formatted Data Units (SFDUs). The format is provided in [10]. The SFDU format is a standard format, based on CCSDS recommendation [11].

7. IMPLEMENTATION STATUS

A prototype has been developed and the implementation work for installing into the DSN has started. The status is discussed below.

7.1 Prototype Status

A two 4-DSP board implementation of the turbo decoder has been developed under the Telecommunications and Mission Operations Directorate (TMOD) Technology (TMOT) program. This decoder has been successfully interfaced with the DSN's Block V Receiver (BVR) in the Telecommunications Development Lab (TDL). The processor speed is 200 MHz. It implements the symbol input, the fractional time tagging, the frame synchronization and six decoder elements. The decoder element speed is about 54 kbps (no stopping rules have been implemented - 10 iterations are done), for an aggregate rate of 324 kbps. Work is under way to implement the stopping rules. It has already reached its initial goal of demonstrating a 250 kbps decoding rate. Preliminary BER testing shows agreement with theory.

7.2 Implementation Plan

The turbo decoder will be installed into the new Downlink Tracking and Telemetry (DTT) subsystem that is being delivered as part of the DSN's Network Simplification Project (NSP). Specifically, the decoder boards will be installed in the new Telemetry Processor (TLP) and controlled by the Downlink Channel Controller (DCC). This equipment will be installed in the DSN in the 2002-2003 time frame.

The decoding capability will be operational at all sites by October 2003. Implementation across the DSN will be staggered over the year 2003, so capability to support missions will be in place at some antennas before October. Full compatibility testing capability will be available in January 2003 and limited compatibility testing will be available prior to that date, using prototype and first production units.

8. CONCLUSION

The turbo decoder being developed at JPL for deep space missions has been described. This decoder allows for missions to use the new CCSDS turbo codes, which provide up to 0.8 dB improvement over the best coding that is currently used. The initial implementation will support at least 365 kbps and will be available for use at all DSN antennas by October 2003.

9. REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," *IEEE International Conference on Communications,* pp. 1064-1070, May 1993.

[2] Consultative Committee for Space Data Systems, Recommendation for Packet Telemetry, CCSDS 102.0-B-4, Blue Book, November 1995.

[3] Consultative Committee for Space Data Systems, Recommendation for Telemetry Channel Coding, CCSDS 101.0-B-4, Blue Book, May 1999.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," IEEE Transactions on Information Theory, IT-20, pp. 284-287, March 1974.

[5] Jeff B. Berner, Selahattin Kayalar, and Jonathan D. Perret, "The NASA Spacecraft Transponding Modem," *2000 IEEE Aerospace Conference Proceedings,* March 18-25, 2000.

[6] K. S. Andrews, J. B. Berner, V. Stanton, V. Chen, and S. Dolinar, "Turbo Decoder Implementation for the DSN," submitted to Telecommunications and Mission Operations Progress Report.

[7] A. Mitache, S. Dolinar, and F. Pollara, "Stopping Rules for Turbo Decoders" Telecommunications and Mission Operations Progress Report 42-142, April-June 2000, pp. 1-22, August 15, 2000.

[8] *Radio Losses for Turbo Codes (8920, 1/3), (8920, 1/6), (1784, 1/3), and (1784, 1/3)*, Shervin Shambayati and Layla Tadjpour, IOM 3311-SS03-00, 6/21/2000, Jet Propulsion Laboratory, Pasadena, CA (an internal document).

[9] Module 207, "34-m and 70-m Telemetry Reception", *Deep Space Mission System Telecommunications Link Design Handbook, Rev. E*, Document 820-005, November

30, 2000, Jet Propulsion Laboratory, Pasadena, CA (an internal document).

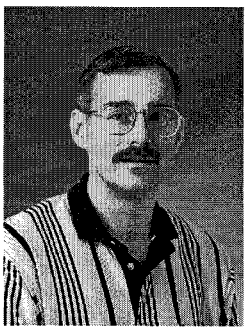
[10] Module TLM-3-29, *Deep Space Mission System External Interface Specification*, Document 820-013, June 15, 2000, Jet Propulsion Laboratory, Pasadena, CA (an internal document).

[11] Consultative Committee for Space Data Systems, Standard Formatted Data Units - Structure and Construction Rules, CCSDS 620.0-B-2, Blue Book, May 1992.

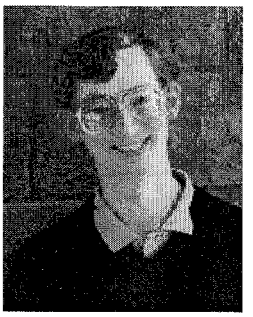
10. ACKNOWLEDGEMENTS

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

We would like to acknowledge all of those who have contributed to the design and implementation of the turbo decoder: Samuel Dolinar, Dariush Divsalar, Fabrizio Pollara, Valerie Stanton, David Wotola, Victor Chen, Jim Weese, Adina Mitache, and Todd Chauvin.



Jeff B. Berner (S'82, M'84, SM'98) earned his B.S. (with honors) in Electrical Engineering in 1983 and his M.S. in Electrical Engineering in 1984, both from the California Institute of Technology. With the exception of a six-month period in 1984, he has been at the Jet Propulsion Laboratory since 1982, where he is a Principal Engineer. Early in his career, he worked on projects such as the Galileo and Mars Observer spacecraft, and the Mobile Satellite Experiment (MSAT-X). More recently, he was the Cognizant Development Engineer of the Block V Receiver, the tracking and telemetry receiver of the Deep Space Network. In 1996, he was awarded the NASA Exceptional Service Medal for his work with the Block V Receiver for the Galileo project. Currently, he is the Telecommunications and Mission Operations Directorate Telecommunications Services System Development Engineer, where he is responsible for the uplink, downlink, and tracking equipment design and performance, and for the STM signal processing architectural design. He is also the task manager for the turbo coding implementation in the DSN.



Kenneth S. Andrews (S'94, M'99) is a member of the Communications Systems and Research Section, Jet Propulsion Laboratory, Pasadena. He received the B. S. degree in Applied Physics from Caltech in 1990 and Ph.D. in Electrical Engineering from Cornell in 1999. From 1990 to 1994, he worked at Altadena Instruments, Pasadena, on spacecraft cameras for the Mars Observer and Mars Global Surveyor missions. His current research work is in error correcting codes and decoder implementations.